Simple Crypto WebGL Templates (v1.4)

Thank you for purchasing my asset and I hope it can get you started developing your NFT integrated game.

Contents

Introduction

This asset is designed to give you a quick start to developing NFT based games by loading and adding the user's owned NFTs into an easily accessible list for you to use however you please. You can choose to use the list for the NFT's Public Address or use the created CryptoNFT ScriptableObject list that is created.

Useful Links

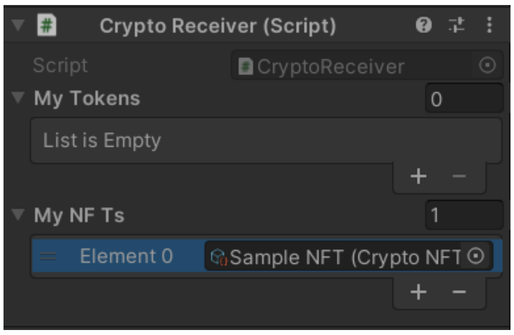
Support Discord -
https://discord.gg/Zbw9dM8KM9
Email -
puppygamingdev@gmail.com
Online Documentation (More Detailed) -
https://puppy-gaming.gitbook.io/simple-crypto-webgl-templates/

Getting Started / Demo

You can easily test the asset works by going to your build settings, adding one of the Demo scenes to the build scenes, then just hit 'Build & Run'. Unity will make the build and launch your browser. Simply wait till the WebGL has loaded and hit the 'Connect' button. Then test it by pressing the 'Update NFTs' button in the game and it should load a new card for each of your owned NFTs. In v1.3 I have also included a 3D Gallery Demo so you can choose which scene to include in your Build to test.
In v1.4, a SOL Transaction Demo is included.
You can see a quick demonstration of the Card Demo at https://youtu.be/uTJJsEFX0ig

Testing In-Editor

I have included a Sample NFT in the Prefabs folder that can be used but you can create your own demo NFT by right clicking in the Project area and going to *'Create > Puppy Gaming > NFT'* and just filling in the Metadata yourself for the new NFT. Then when your game is running in the editor select the CryptoReceiver GameObject in the hierarchy (nested under DontDestroyOnLoad), add a space in the My NFTs list and drag your test NFT to it.



Crypto Receiver (Script)

Script	CryptoReceiver
My Tokens	0
List is Empty	
My NF Ts	1
Element 0	Sample NFT (Crypto NFT

Workflow / Script References

<u>Puppy Gaming/Scripts/Plugins/solana.jslib</u>
This file contains all the code needed for the WebGL player and also the code needed to log into the wallet from the selected chain, then loading all the wallets NFTs and finally grabbing the metadata from them and sending them to the CryptoReceiver GameObject in your scene. The functions in this file are commented on for easier understanding.

<u>Puppy Gaming/Prefabs/CryptoReceiver.cs</u>
The CryptoReceiver prefab should be placed in your first Scene when the player would be logging into their wallet, it will persist across scenes so no need to add it anywhere else.
The CryptoReceiver script receives the data from the Login request process and puts the metadata from each NFT into a CryptoNFT ScriptableObject and then instantiates that object in the MyNFTs List from the *ReceiveNFT()* function. This list can be accessed by any other script with *CryptoReceiver.CR.myNFTs*
The current accessible data from a CryptoNFT object is;
.nftName - The NFTs name,
.spritelink - The image URL for the NFT,
.description - The collections description,
.attributes - A List of attribute scriptable objects for each attribute.
It also creates a list of NFT Public Addresses that can be found in *CryptoReceiver.CR.myNFTsList* which helps if you don't need to grab metadata and just want to use the Addresses for comparisons.

You can get the user's Wallet Address from *CryptoReceiver.CR.walletAddress* and a shortened version of the address for in-game display like 'AaAa….BbBb' from *CryptoReceiver.CR.shortAddress*
You can check whether the address has been collected with *CryptoReceiver.CR.isConnected* bool
From version 1.4 it now also receives a list of SPL tokens, with an amount and also the decimals the token uses.

Provided Demos

<u>Card Demo</u>
This demo provides a sample of how to instantiate cards into your scene from the NFTPrefab objects in the *CryptoReceiver.CR.myNFTs* List.
Feel free to reuse any of the code from it or just edit the scene to your liking.
From version 1.4 this had been modified for the 'card' to cover a larger area of the screen to also display an NFT's attributes.

<u>Gallery Demo</u>
The same applies from the Card Demo, but the NFTs will be instantiated as art works in an art 3D art gallery environment (apologies for the poor character controller).

<u>SOL Transaction Demo</u>
This demo simply supplies a showcase and example on how to use the SOL Transactions added in version 1.4, The Cluster is set to Devnet by default and you can change the destination address by changing the text of it. For using the function in your own projects, you can pass it as a string to the function, no need to have it written there but WebGL in 2021 lacks support for Copy/Paste from the System Clipboard. There are 3rd party assets / repos that you can use to resolve this but I cannot supply them in my asset.